ソフトウェア進化と自己適応ソフトウェア

研究ノート

中　川　博　之*

Software evolution and self-adaptive software

Key Words：software evolution, self-adaptive software, self-adaptive systems, requirements engineering

1．はじめに

近年，ソフトウェアの活躍する場面が広がり，ソフトウェアシステムは社会インフラとしてだけでなく，組込みシステムや携帯端末，Webアプリケーションのように我々の身近においても生活を支える存在となっている．これらのソフトウェアの多くは実世界の事象を扱うものであり，多様な環境下での動作や，ユーザ要求の変化への柔軟な対応が期待されている．

従来より，ソフトウェアの振る舞い（仕様）とソフトウェアに対する要求については，ZaveとJacksonの以下の関係式が知られている[1]．

$$S, W \vdash R$$

ここで，Sは構築するソフトウェアの振る舞いであり，Wはソフトウェアを取り巻く環境の状態を，Rはソフトウェアに対する要求を指す．つまり，ソフトウェアは，環境Wに対して，求められる状態Rに到達する（Rを導出できる）ような状態遷移を与える振る舞いSを持つべきであることを意味している．

近年では，ソフトウェアが置かれる環境の多様化（つまりWの変化）やユーザ要求の変化（Rの変化）への柔軟な対応が求められており，ZaveとJacksonの関係式を拡張してソフトウェアを扱うことが必要となってきている[2]．たとえば，Rの変化（$R \Rightarrow R'$）に対して，R'を満足するようにSをS'に置き換えるソフトウェアの変更行為は，ソフトウェア進化（software evolution）と呼ばれている（$S, W \vdash R \Rightarrow S', W \vdash R'$）．ソフトウェア進化とは，ソフトウェア保守行為の一種であるが，従来のバグ対応のような受動的な保守ではなく，機能追加などの積極的な変更を指すために近年特に着目されている行為である．一方，Wの変化（$W \Rightarrow W'$）に対して，W'のもとでもRを達成できるように振る舞いをSからS'に変化させる行為は，適応（adaptation）と呼ばれる（$S, W \vdash R \Rightarrow S', W' \vdash R$）．適応に対しては，開発者が介在することなく，Wの変化をソフトウェア自身が検知し，自ら振る舞いを変化させることで環境変化に対応することが求められている．このようなソフトウェアは自己適応ソフトウェア（self-adaptive software），あるいは自己適応システム（self-adaptive systems）と呼ばれ，近年その実現方法が活発に研究されている．本研究ノートでは，このソフトウェア進化と自己適応ソフトウェアに対する筆者の取り組みを紹介する．

2．アプローチ

ソフトウェア進化や適応を扱う場合，ZaveとJacksonの関係式におけるSをS'に置き換える行為が必要となる．ここで重要なのは，単に変更後のS'だけを考えるのではなく，既存のSに対して如何に小さい差分（ΔS）によりS'を用意できるかということである（$S + \Delta S \Rightarrow S'$）．このためには，Sに対する変更影響の局所化が有効である．

筆者は，Sの変更影響を局所化するために，Sの構成要素を独立性の高いモジュール，つまり依存関係を極力排除したモジュールとして設計することを考え，制御モデルとして知られるControl loop[3]に着目した．Control loopとは，環境情報の収集

* Hiroyuki NAKAGAWA
1974年12月生
東京大学大学院情報科学研究科　博士課程中退（2008年）
現在、大阪大学　大学院情報科学研究科　情報システム学専攻　ディペンダビリティ工学講座　准教授　博士（工学）
ソフトウェア工学
TEL：06-6879-4536
FAX：06-6879-4539
E-mail：nakagawa@ist.osaka-u.ac.jp

（Collect），状況分析（Analyze），次に実行すべきアクションの決定（Decide），アクション実行（Act）の4つの制御動作を独立して提供可能な実行単位であり，独立性の高いControl loopを複数組み合わせてシステムを構築することにより，Sにおける変更影響の局所化が期待できる．

3．ゴールモデルの整形

では，どのようにSの構成要素，つまりControl loop群を決定すれば良いであろうか？ソフトウェア進化の場合にはRが，自己適応ソフトウェア（適応）の場合にはWがSの変更を駆動するといえる．従って，筆者はRとWの記述を整形することでControl loop群を決定する手法を考案した[4]．この手法では，RとWを要求記述モデルとして知られるゴールモデル上に記述し，ゴールモデルを提案する整形手法に従って変形することで，複数のControl loopをゴールモデルから抽出する．ゴールモデルはシステムに対する要求を記述し，システムが達成すべき状態に順次分解（洗練化と呼ばれている）することで，要求と状態とを体系的に構造化するモデルである．ゴールモデル整形により抽出されたControl loopはSの構成要素となる．

図1はゴールモデル整形プロセスの概要である．本整形プロセスでは，Step2でControl loopを形成可能な箇所を判断し，Step3で図2に示したControl loopパターンに合致するようにゴールモデルを整形する．これによりゴールモデル中にControl loopが複数出現することになるが，Step4により

Control loop間で発生し得る競合をControl loopの統合や追加により解消することで，最終的に該当ソフトウェアに必要なControl loop群を同定する．例えば、図3は清掃ロボット制御ソフトウェアのゴールモデルに対する整形プロセスの適用結果を示したものである．図3からは，本ソフトウェアは3つのControl loopにより構成すればよいことがわかる．

ソフトウェア進化を考える場合，要求の変化（R⇒R'）が前提にあるが，RとR'に対してそれぞれゴールモデルを構築し，整形後に得られるS，S'の差分（各Control loop群の差分）を取ることで，変更影響が局所化されたΔSを得ることができる．一方，自己適応ソフトウェアに対しては，想定し得る環境状態を含んだゴールモデルからSやS'に対応するControl loop群を決定しておくことで，適応時の変更影響を限定化することが可能となる．

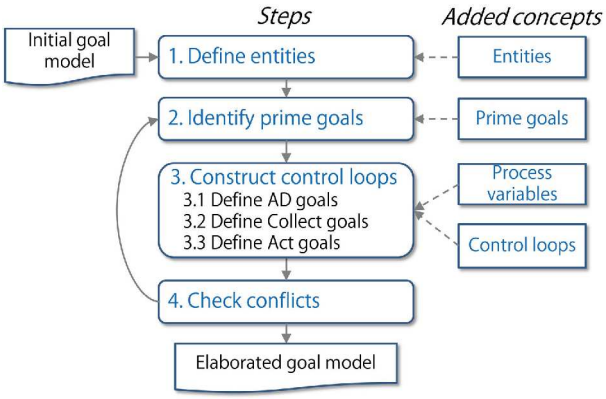

図1　ゴールモデル整形プロセスとControl loopパターン

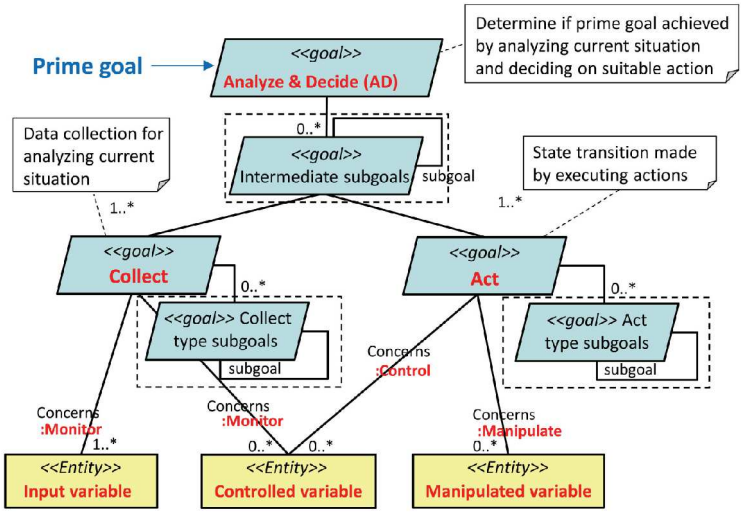

図2　Control loopパターン

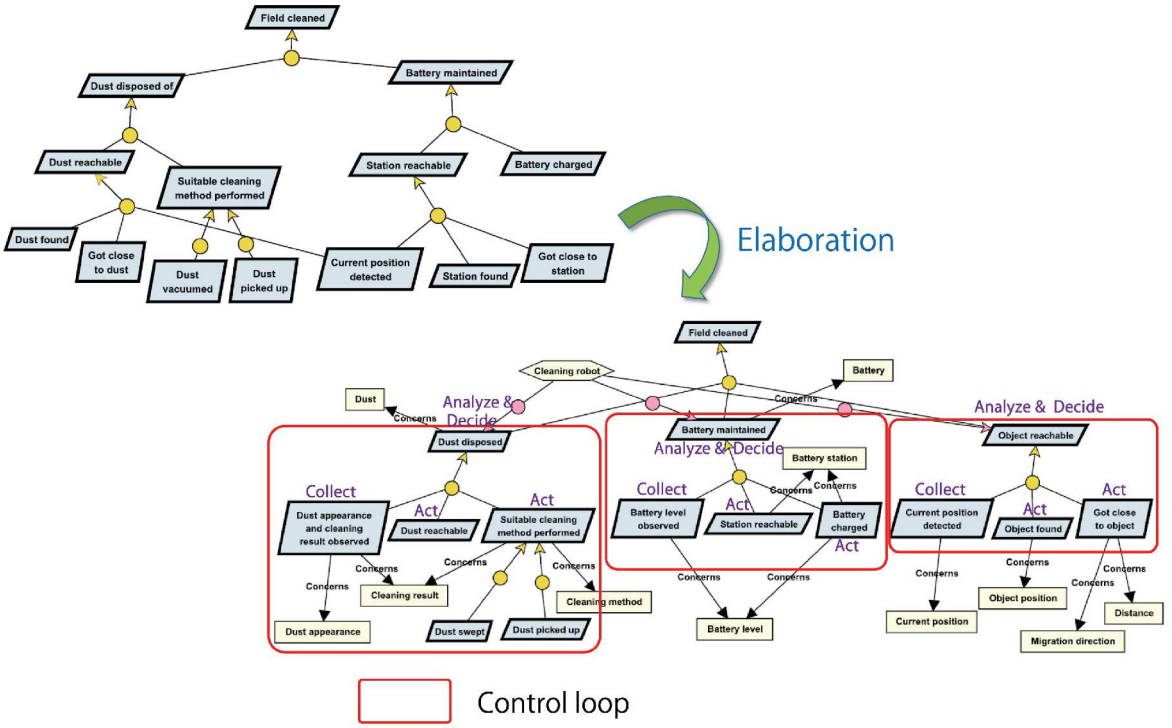

図3　整形プロセスの適用例

4．開発支援ツール

Control loop群に基づいたソフトウェアの開発プロセスを図4に示す．本開発プロセスでは大きく2つの開発支援ツールを利用する．まず，ゴールモデルコンパイラ [5][6] に整形後のゴールモデルを与えることで，Control loop実装に関する情報（プログラムのテンプレートや接続情報など）と，差分情報などのソフトウェア進化を支援する情報を取得することができる（図4中のGoal model compilerから出力される各モデル）．また，各Control loopの実装と，実装後のデプロイ（配備），実行制御には，プログラミングフレームワーク [6] を利用する．このプログラミングフレームワークの特徴の一つとして，ダイナミックデプロイが挙げられる．これは，ソフトウェア実行中にもControl loopを配備，制御できるメカニズムであり，これをシステム自身が駆動するよう拡張することで，自己適応ソフトウェアの適応メカニズムを実現することも可能となる．

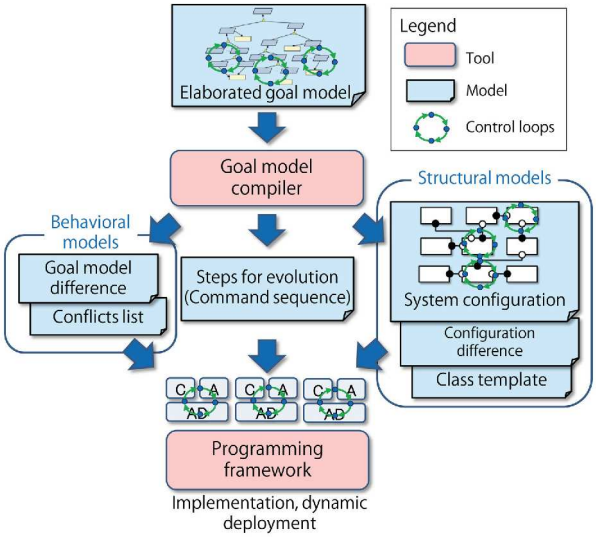

図4　開発プロセスの概要

5．おわりに

本稿では，近年ソフトウェア工学分野で注目されているソフトウェア進化と自己適応ソフトウェアについて，筆者の取り組みを概説した．近年では特に

Webアプリケーション分野において継続的なソフトウェア進化（continuous software evolution）が望まれており，本研究のような影響範囲を限定する手法が大いに期待されている．一方，自己適応ソフトウェアはソフトウェアの新たな可能性を追及する試みであり，ここ数年間で研究対象が急激に広域化，深化しているところである．いずれも今後の研究成果から目が離せない．

参考文献

[1] P. Zave and M. Jackson, “Four dark corners of requirements engineering,” ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 6, no. 1, pp. 1－30, Jan. 1997.

[2] R. Calinescu, C. Ghezzi, M. Kwiatkowska, and R. Mirandola, “Self-adaptive software needs quantitative verification at runtime”, Communications of the ACM, vol. 55, No. 9, pp. 69－77, Sep, 2012.

[3] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli, “A survey of autonomic communications,” ACM Transactions on Autonomous and Adaptive Systems (TAAS), vol. 1, no. 2, pp. 223－259, 2006.

[4] H. Nakagawa, A. Ohsuga, and S. Honiden, “A goal model elaboration for localizing changes in software evolution,” in Proc. of the 21st IEEE International Requirements Engineering Conference (RE’13), pp. 155－164, IEEE CS, Jul, 2013.

[5] H. Nakagawa, A. Ohsuga, and S. Honiden, “gocc: A configuration compiler for self-adaptive systems using goal-oriented requirements description,” in Proc. of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS’11), pp. 40－49, ACM, May, 2011.

[6] H. Nakagawa, A. Ohsuga, and S. Honiden, “Towards dynamic evolution of self-adaptive systems based on dynamic updating of control loops,” in Proc. of the sixth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO’12), pp. 59－68, IEEE, Sep, 2012.